```
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                AAAAAAAAA     TTTTTTTTTTTTTTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM  MMM MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM  MMM MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM  MMM MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEEEEEEEEEEE     MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEEEEEEEEEEE     MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEEEEEEEEEEE     MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAAAAAAAAAAAAAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAAAAAAAAAAAAAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAAAAAAAAAAAAAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEE              MMM      MMM  MMM UUU         UUU  LLL              AAA         AAA       TTT
EEEEEEEEEEEEEEE  MMM      MMM  MMM UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA       TTT
EEEEEEEEEEEEEEE  MMM      MMM  MMM UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA       TTT
EEEEEEEEEEEEEEE  MMM      MMM  MMM UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA       TTT
```

FILEID**BOOSTRING

```
BBBBBBBB      OOOOOO      OOOOOO      SSSSSSSS  TTTTTTTTTT  RRRRRRRR      IIIIII    NN      NN    GGGGGGGG
BBBBBBBB      OOOOOO      OOOOOO      SSSSSSSS  TTTTTTTTTT  RRRRRRRR      IIIIII    NN      NN    GGGGGGGG
BB      BB  OO      OO  OO      OO  SS            TT      RR      RR      II      NN      NN  GG
BB      BB  OO      OO  OO      OO  SS            TT      RR      RR      II      NN      NN  GG
BB      BB  OO      OO  OO      OO  SS            TT      RR      RR      II      NNNN    NN  GG
BB      BB  OO      OO  OO      OO  SS            TT      RR      RR      II      NNNN    NN  GG
BBBBBBBB    OO      OO  OO      OO    SSSSSS      TT      RRRRRRRR        II      NN  NN  NN  GG
BBBBBBBB    OO      OO  OO      OO    SSSSSS      TT      RRRRRRRR        II      NN  NN  NN  GG
BB      BB  OO      OO  OO      OO          SS    TT      RR  RR          II      NN    NNNN  GG  GGGGGG
BB      BB  OO      OO  OO      OO          SS    TT      RR  RR          II      NN    NNNN  GG  GGGGGG
BB      BB  OO      OO  OO      OO          SS    TT      RR    RR        II      NN      NN  GG      GG      ....
BB      BB  OO      OO  OO      OO          SS    TT      RR    RR        II      NN      NN  GG      GG      ....
BBBBBBBB      OOOOOO      OOOOOO    SSSSSSSS      TT      RR      RR    IIIIII    NN      NN    GGGGGG        ....
BBBBBBBB      OOOOOO      OOOOOO    SSSSSSSS      TT      RR      RR    IIIIII    NN      NN    GGGGGG        ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
00000001  0000     1        BOOT_SWITCH = 1                 ; Include bootstrap emulation subset
```

```
0000     1          .NOSHOW CONDITIONALS
0000     5          .TITLE  BOO$STRING      Subset Instruction Emulation for VMB and SYSBOOT
0000     7          .IDENT  /V04-001/
0000     8
0000     9 ;
0000    10 ;*****************************************************************************
0000    11 ;*                                                                           *
0000    12 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    13 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    14 ;*  ALL RIGHTS RESERVED.                                                     *
0000    15 ;*                                                                           *
0000    16 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    17 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    18 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    19 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    20 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    21 ;*  TRANSFERRED.                                                             *
0000    22 ;*                                                                           *
0000    23 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    24 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    25 ;*  CORPORATION.                                                             *
0000    26 ;*                                                                           *
0000    27 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    28 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    29 ;*                                                                           *
0000    30 ;*                                                                           *
0000    31 ;*****************************************************************************
0000    32 ;
0000    33
0000    34 ;++
0000    35 ; Facility:
0000    36 ;
0000    37 ;       VAX-11 Instruction Emulator
0000    38 ;
0000    39 ; Abstract:
0000    40 ;
0000    41 ;       The routines in this module emulate the VAX-11 string instructions.
0000    42 ;       These procedures can be a part of an emulator package or can be
0000    43 ;       called directly after the input parameters have been loaded into
0000    44 ;       the architectural registers.
0000    45 ;
0000    46 ;       The input parameters to these routines are the registers that
0000    47 ;       contain the intermediate instruction state.
0000    48 ;
0000    49 ; Environment:
0000    50 ;
0000    51 ;       These routines run at any access mode, at any IPL, and are AST
0000    52 ;       reentrant.
0000    53 ;
0000    54 ; Author:
0000    55 ;
0000    56 ;       Lawrence J. Kenah
0000    57 ;
0000    58 ; Creation Date:
0000    59 ;
0000    60 ;       16 August 1982
0000    61 ;
```

```
0000     62 ; Modified by:
0000     63 ;
0000     64 ;       V04-001 LJK0044         Lawrence J. Kenah       6-Sep-1984
0000     65 ;               The backup code for MOVTC when moving in the forward direction
0000     66 ;               also needs to be changed (see LJK0039) based on the relative
0000     67 ;               sizes of the source and destination strings.
0000     68 ;
0000     69 ;       V01-005 KDM0107         Kathleen D. Morse       21-Aug-1984
0000     70 ;               Fix bug in CMPC3.  Return C clear if string length is 0.
0000     71 ;
0000     72 ;       V01-004 LJK0039         Lawrence J. Kenah       20-Jul-1984
0000     73 ;               Mofify MOVTC backup code to reflect differences in register
0000     74 ;               contents when traversing strings backwards. There are two
0000     75 ;               cases based on the relative sizes of source and destination.
0000     76 ;
0000     77 ;       V01-003 LJK0026         Lawrence J. Kenah       19-Mar-1984
0000     78 ;               Final cleanup pass. Access violation handler is now called
0000     79 ;               STRING_ACCVIO. Set PACK_M_ACCVIO bit in R1 before passing
0000     80 ;               control to VAX$REFLECT_FAULT.
0000     81 ;
0000     82 ;       V01-002 LJK0011         Lawrence J. Kenah       8-Nov-1983
0000     83 ;               Fix three minor bugs in MOVTC and MOVTUC. Change exception
0000     84 ;               handling to reflect chenged implementation.
0000     85 ;
0000     86 ;       V01-001 Original        Lawrence J. Kenah       16-Aug-1982
0000     87 ;--
```

```
0000     89          .SUBTITLE       Miscellaneous Notes
0000     90 ;+
0000     91 ;       The following notes apply to most or all of the routines that appear in
0000     92 ;       this module. The comments appear here to avoid duplication in each routine.
0000     93 ;
0000     94 ;  1.   The VAX Architecture Standard (DEC STD 032) is the ultimate authority on
0000     95 ;       the functional behavior of these routines. A summary of each instruction
0000     96 ;       that is emulated appears in the Functional Description section of each
0000     97 ;       routine header.
0000     98 ;
0000     99 ;  2.   One design goal that affects the algorithms used is that these instructions
0000    100 ;       can incur exceptions such as access violations that will be reported to
0000    101 ;       users in such a way that the exception appears to have originated at the
0000    102 ;       site of the reserved instruction rather than within the emulator. This
0000    103 ;       constraint affects the algorithms available and dictates specific
0000    104 ;       implementation decisions.
0000    105 ;
0000    106 ;  3.   Each routine header contains a picture of the register usage when it is
0000    107 ;       necessary to store the intermediate state of an instruction (routine) while
0000    108 ;       servicing an exception.
0000    109 ;
0000    110 ;       The delta-PC field is used by the condition handler jacket to these
0000    111 ;       routines when it determines that an exception such as an access violation
0000    112 ;       occurred in response to an explicit use of one of the reserved
0000    113 ;       instructions. These routines can also be called directly with the input
0000    114 ;       parameters correctly placed in registers. The delta-PC field is not used in
0000    115 ;       this case.
0000    116 ;
0000    117 ;       Note that the input parameters to any routine are a subset of the
0000    118 ;       intermediate state picture.
0000    119 ;
0000    120 ;       Fields that are not used either as input parameters or to store
0000    121 ;       intermediate state are indicated thus, XXXXX.
0000    122 ;
0000    123 ;  4.   In the Input Parameter list for each routine, certain register fields that
0000    124 ;       are not used may be explicitly listed for one reason or another. These
0000    125 ;       unused input parameters are described as IRRELEVANT.
0000    126 ;
0000    127 ;  5.   In general, the final condition code settings are determined as the side
0000    128 ;       effect of one of the last instructions that executes before control is
0000    129 ;       passed back to the caller with an RSB. It is seldom necessary to explicitly
0000    130 ;       manipulate condition codes with a BIxPSW instruction or similar means.
0000    131 ;
0000    132 ;  6.   There is only a small set of exceptions that are reflected to the user in an
0000    133 ;       altered fashion, with the exception PC changed from within the emulator to
0000    134 ;       the site of the original entry into these routines. The instructions that
0000    135 ;       generate these exceptions are all immediately preceded by a
0000    136 ;
0000    137 ;               MARK_POINT      yyyy_N
0000    138 ;
0000    139 ;       where yyyy is the instruction name and N is a small integer. These names
0000    140 ;       map directly into instruction- and context-specific routines (located at
0000    141 ;       the end of this module) that put each instruction (routine) into a
0000    142 ;       consistent state before passing control to a more general exception handler
0000    143 ;       in a different module.
0000    144 ;-
```

```
                        .SUBTITLE       DECLARATIONS
                        
                ; Include files:
                
                        $PSLDEF                         ; Define bit fields in PSL
                
                        .NOCROSS                        ; No cross reference for these
                        .ENABLE         SUPPRESSION     ; No symbol table entries either
                
                        PACK_DEF                        ; Stack usage for exception handling
                
                        .DISABLE        SUPPRESSION     ; Turn on symbol table again
                        .CROSS                          ; Cross reference is OK now
                
                ; Macro Definitions
                
                        .MACRO  _INCLUDE        OPCODE , BOOT_FLAG
                        .IF     NOT_DEFINED     BOOT_SWITCH
                                OPCODE'_DEF
                                INCLUDE_'OPCODE = 0
                        .IF_FALSE
                                .IF     IDENTICAL       <BOOT_FLAG> , BOOT
                                        OPCODE'_DEF
                                        INCLUDE_'OPCODE = 0
                                .ENDC
                        .ENDC
                        .ENDM   _INCLUDE
                
                ; External declarations
                
                        .DISABLE        GLOBAL
                
                
                ; PSECT Declarations:
                
                        .DEFAULT        DISPLACEMENT , WORD
                
00000000                .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, LONG
                
                        BEGIN_MARK_POINT                ; Set up exception mark points
```

```
0000    190          .SUBTITLE       Conditional Assembly Parameters
0000    191 ;+
0000    192 ; Functional Description:
0000    193 ;
0000    194 ;       It is possible to create a subset emulator, one that emulates
0000    195 ;       specific reserved instructions. This capability is currently exploited
0000    196 ;       to create a subset emulator for use by the bootstrap programs.
0000    197 ;
0000    198 ;       An instruction is included in the full emulator by making an entry
0000    199 ;       in the following table. If the optional second parameter is present
0000    200 ;       and equal to BOOT, then that instruction is included in the subset
0000    201 ;       emulator used by the bootstrap code.
0000    202 ;-
0000    203
0000    204          .NOCROSS                           ; No cross reference for these
0000    205          .ENABLE         SUPPRESSION        ; No symbol table entries either
0000    206
0000    207          _INCLUDE        MOVTC
0000    208          _INCLUDE        MOVTUC
0000    209          _INCLUDE        CMPC3 , BOOT
0000    210          _INCLUDE        CMPC5 , BOOT
0000    211          _INCLUDE        SCANC
0000    212          _INCLUDE        SPANC
0000    213          _INCLUDE        LOCC , BOOT
0000    214          _INCLUDE        SKPC
0000    215          _INCLUDE        MATCHC
0000    216          _INCLUDE        CRC
0000    217
0000    218          .DISABLE        SUPPRESSION        ; Turn on symbol table again
0000    219          .CROSS                             ; Cross reference is OK now
0000    220
0000    221          .NOSHOW         CONDITIONALS
0000    222
```

```
0000    683          .SUBTITLE       VAX$CMPC3 - Compare Characters (3 Operand)
0000    684 ;+
0000    685 ; Functional Description:
0000    686 ;
0000    687 ;       The bytes of string 1 specified by the length and address 1 operands are
0000    688 ;       compared  with the bytes of string 2 specified by the length and address
0000    689 ;       2 operands.  Comparison proceeds until inequality is detected or all the
0000    690 ;       bytes  of  the strings have been examined.  Condition codes are affected
0000    691 ;       by the result of the last byte  comparison.   Two  zero  length  strings
0000    692 ;       compare equal (i.e.  Z is set and N, V, and C are cleared).
0000    693 ;
0000    694 ; Input Parameters:
0000    695 ;
0000    696 ;       R0<15:0> = len          Length of character strings
0000    697 ;       R1       = src1addr     Address of first character string (called S1)
0000    698 ;       R3       = src2addr     Address of second character string (called S2)
0000    699 ;
0000    700 ; Intermediate State:
0000    701 ;
0000    702 ;        31               23               15               07              00
0000    703 ;       +----------------+----------------+----------------+----------------+
0000    704 ;       |    delta-PC    |      XXXX      |                len             | : R0
0000    705 ;       +----------------+----------------+----------------+----------------+
0000    706 ;       |                              src1addr                             | : R1
0000    707 ;       +----------------+----------------+----------------+----------------+
0000    708 ;       |                                XXXXX                              | : R2
0000    709 ;       +----------------+----------------+----------------+----------------+
0000    710 ;       |                              src2addr                             | : R3
0000    711 ;       +----------------+----------------+----------------+----------------+
0000    712 ;
0000    713 ; Output Parameters:
0000    714 ;
0000    715 ;       Strings are IDENTICAL
0000    716 ;
0000    717 ;               R0 = 0
0000    718 ;               R1 = Address of one byte beyond end of S1
0000    719 ;               R2 = 0 (same as R0)
0000    720 ;               R1 = Address of one byte beyond end of S2
0000    721 ;
0000    722 ;       Strings DO NOT MATCH
0000    723 ;
0000    724 ;               R0 = Number of bytes left in strings (including first byte
0000    725 ;                    that did not match)
0000    726 ;               R1 = Address of nonmatching byte in S1
0000    727 ;               R2 = R0
0000    728 ;               R3 = Address of nonmatching byte in S2
0000    729 ;
0000    730 ; Condition Codes:
0000    731 ;
0000    732 ;       In general, the condition codes reflect whether or not the strings
0000    733 ;       are considered the same or different. In the case of different
0000    734 ;       strings, the condition codes reflect the result of the comparison
0000    735 ;       that indicated that the strings are not equal.
0000    736 ;
0000    737 ;       Strings are IDENTICAL
0000    738 ;
0000    739 ;               N <- 0
```

```
                     0000   740 ;                  Z <- 1                          ; (byte in S1) EQL (byte in S2)
                     0000   741 ;                  V <- 0
                     0000   742 ;                  C <- 0
                     0000   743 ;
                     0000   744 ;       Strings DO NOT MATCH
                     0000   745 ;
                     0000   746 ;                  N <- (byte in S1) LSS (byte in S2)
                     0000   747 ;                  Z <- 0                          ; (byte in S1) NEQ (byte in S2)
                     0000   748 ;                  V <- 0
                     0000   749 ;                  C <- (byte in S1) LSSU (byte in S2)
                     0000   750 ;
                     0000   751 ;       where "byte in S1" or "byte in S2" may indicate the fill character
                     0000   752 ;
                     0000   753 ; Side Effects:
                     0000   754 ;
                     0000   755 ;       This routine uses one longword of stack.
                     0000   756 ;-
                     0000   757
                     0000   758 VAX$CMPC3::
 50    50    3C      0000   759          MOVZWL  R0,R0                           ; Clear unused bits & check for zero
       0D    13      0003   760          BEQL    20$                             ; Simply return if zero length string
                     0005   761
       5A    DD      0005   762          PUSHL   R10                             ; Save R10 so it can hold handler
                     0007   763          ESTABLISH_HANDLER       -
                     0007   764                  STRING_ACCVIO                   ; Store address of condition handler
                     0007   765
                     0007   766          MARK_POINT      CMPC3_1
 81    83    91      0007   767 10$:     CMPB    (R3)+,(R1)+                     ; Character match?
       0B    12      000A   768          BNEQ    30$                             ; Exit loop if different
     F8 50   F5      000C   769          SOBGTR  R0,10$
                     000F   770
                     000F   771 ; Exit path for strings IDENTICAL (R0 = 0, either on input or after loop)
                     000F   772
 5A    8E    D0      000F   773          MOVL    (SP)+,R10                       ; Restore saved R10
       52    D4      0012   774 20$:     CLRL    R2                              ; Set R2 for output value of 0
       50    D5      0014   775          TSTL    R0                              ; Set condition codes
             05      0016   776          RSB                                     ; Return point for IDENTICAL strings
                     0017   777
                     0017   778 ; Exit path when strings DO NOT MATCH
                     0017   779
 5A    8E    D0      0017   780 30$:     MOVL    (SP)+,R10                       ; Restore saved R10
 52    50    D0      001A   781          MOVL    R0,R2                           ; R0 and R2 are the same on exit
 73    71    91      001D   782          CMPB    -(R1),-(R3)                     ; Reset R1 and R3 and set condition codes
             05      0020   783          RSB                                     ; Return point when strings DO NOT MATCH
```

```
0021    787          .SUBTITLE       VAX$CMPC5 - Compare Characters (5 Operand)
0021    788 ;+
0021    789 ; Functional Description:
0021    790 ;
0021    791 ;       The bytes of the string 1 specified  by  the  length  1  and  address  1
0021    792 ;       operands  are  compared  with the bytes of the string 2 specified by the
0021    793 ;       length 2 and address 2 operands.  If  one  string  is  longer  than  the
0021    794 ;       other,  the shorter string is conceptually extended to the length of the
0021    795 ;       longer by appending (at  higher  addresses)  bytes  equal  to  the  fill
0021    796 ;       operand.   Comparison  proceeds  until inequality is detected or all the
0021    797 ;       bytes of the strings have been examined.  Condition codes  are  affected
0021    798 ;       by  the  result  of  the  last byte comparison.  Two zero length strings
0021    799 ;       compare equal (i.e.  Z is set and N, V, and C are cleared).
0021    800 ;
0021    801 ; Input Parameters:
0021    802 ;
0021    803 ;       R0<15:0>  = len         Length of first character string (called S1)
0021    804 ;       R0<23:16> = fill        Fill character that is used when strings have
0021    805 ;                                 different lengths
0021    806 ;       R1        = addr        Address of first character string
0021    807 ;       R2<15:0>  = len         Length of second character string (called S2)
0021    808 ;       R3        = addr        Address of second character string
0021    809 ;
0021    810 ; Intermediate State:
0021    811 ;
0021    812 ;        31              23              15              07              00
0021    813 ;       +---------------+---------------+---------------+---------------+
0021    814 ;       |    delta-PC   |      fill     |             src1len           | : R0
0021    815 ;       +---------------+---------------+---------------+---------------+
0021    816 ;       |                             src1addr                          | : R1
0021    817 ;       +---------------+---------------+---------------+---------------+
0021    818 ;       |               XXXXX           |             src2len           | : R2
0021    819 ;       +---------------+---------------+---------------+---------------+
0021    820 ;       |                             src2addr                          | : R3
0021    821 ;       +---------------+---------------+---------------+---------------+
0021    822 ;
0021    823 ; Output Parameters:
0021    824 ;
0021    825 ;       Strings are IDENTICAL
0021    826 ;
0021    827 ;               R0 = 0
0021    828 ;               R1 = Address of one byte beyond end of S1
0021    829 ;               R2 = 0 (same as R0)
0021    830 ;               R1 = Address of one byte beyond end of S2
0021    831 ;
0021    832 ;       Strings DO NOT MATCH
0021    833 ;
0021    834 ;               R0 = Number of bytes remaining in S1 when mismatch detected
0021    835 ;                    (or zero if S1 exhausted before mismatch detected)
0021    836 ;               R1 = Address of nonmatching byte in S1
0021    837 ;               R2 = Number of bytes remaining in S2 when mismatch detected
0021    838 ;                    (or zero if S2 exhausted before mismatch detected)
0021    839 ;               R3 = Address of nonmatching byte in S2
0021    840 ;
0021    841 ; Condition Codes:
0021    842 ;
0021    843 ;       In general, the condition codes reflect whether or not the strings
```

```
                     0021   844 ;       are considered the same or different. In the case of different
                     0021   845 ;       strings, the condition codes reflect the result of the comparison
                     0021   846 ;       that indicated that the strings are not equal.
                     0021   847 ;
                     0021   848 ;       Strings are IDENTICAL
                     0021   849 ;
                     0021   850 ;               N <- 0
                     0021   851 ;               Z <- 1                  ; (byte in S1) EQL (byte in S2)
                     0021   852 ;               V <- 0
                     0021   853 ;               C <- 0
                     0021   854 ;
                     0021   855 ;       Strings DO NOT MATCH
                     0021   856 ;
                     0021   857 ;               N <- (byte in S1) LSS (byte in S2)
                     0021   858 ;               Z <- 0                  ; (byte in S1) NEQ (byte in S2)
                     0021   859 ;               V <- 0
                     0021   860 ;               C <- (byte in S1) LSSU (byte in S2)
                     0021   861 ;
                     0021   862 ;       where "byte in S1" or "byte in S2" may indicate the fill character
                     0021   863 ;
                     0021   864 ; Side Effects:
                     0021   865 ;
                     0021   866 ;       This routine uses two longwords of stack.
                     0021   867 ;-
                     0021   868
                     0021   869         .ENABLE LOCAL_BLOCK
                     0021   870
                     0021   871 VAX$CMPC5::
                5A   DD 0021   872         PUSHL   R10                     ; Save R10 so it can hold handler
                        0023   873         ESTABLISH_HANDLER       -
                        0023   874                 STRING_ACCVIO           ; Store address of condition handler
                     54 DD 0023   875         PUSHL   R4                      ; Save register
             54      50 FO 8F   78 0025   876         ASHL    #-16,R0,R4              ; Get escape character
                  50    50   3C 002A   877         MOVZWL  R0,R0                   ; Clear unused bits & is S1 length zero?
                        28   13 002D   878         BEQL    50$                     ; Branch if yes
                     52 52   3C 002F   879         MOVZWL  R2,R2                   ; Clear unused bits & is S2 length zero?
                        14   13 0032   880         BEQL    30$
                             0034   881
                             0034   882 ; Main loop. The following loop executes when both strings have characters
                             0034   883 ; remaining and inequality has not yet been detected.
                             0034   884
                             0034   885 ; THE FOLLOWING LOOP IS A TARGET FOR FURTHER OPTIMIZATION IN THAT THE
                             0034   886 ; LOOP SHOULD NOT REQUIRE TWO SOBGTR INSTRUCTIONS. NOTE, THOUGH, THAT
                             0034   887 ; THE CURRENT UNOPTIMIZED LOOP IS EASIER TO BACK UP.
                             0034   888
                             0034   889         MARK_POINT      CMPC5_1
                     83 81   91 0034   890 10$:    CMPB    (R1)+,(R3)+             ; Characters match?
                        32   12 0037   891         BNEQ    80$                     ; Exit loop if bytes different
                  09 50      F5 0039   892         SOBGTR  R0,20$                  ; Check for S1 exhausted
                             003C   893
                             003C   894 ; The next test determines whether S2 is also exhausted.
                             003C   895
                     52 D7   003C   896         DECL    R2                      ; Put R2 in step with R0
                     1C 12   003E   897         BNEQ    60$                     ; Branch if bytes remaining in S2
                             0040   898
                             0040   899 ; This is the exit path for identical strings. If we get here, then both
                             0040   900 ; R0 and R2 are zero.  The condition codes are correctly set (by the ASHL
```

```
                     0040   901 ; instruction) so the registers are restored with a POPR to avoid changing
                     0040   902 ; the condition codes.
                     0040   903
                     0040   904 IDENTICAL:
0410 8F    BA        0040   905         POPR    #^M<R4,R10>                     ; Restore saved registers
           05        0044   906         RSB                                     ; Exit indicating IDENTICAL strings
                     0045   907
  EC 52    F5        0045   908 20$:    SOBGTR  R2,10$                          ; Check for S2 exhausted
                     0048   909
                     0048   910 ; The following loop is entered when all of S2 has been processed but
                     0048   911 ; there are characters remaining in S1. In other words,
                     0048   912 ;
                     0048   913 ;       R0 GTRU 0
                     0048   914 ;       R2 EQL 0
                     0048   915 ;
                     0048   916 ; The remaining characters in S1 are compared to the fill character.
                     0048   917
                     0048   918         MARK_POINT      CMPC5_2
54   81    91        0048   919 30$:    CMPB    (R1)+,R4                        ; Characters match?
      05   12        004B   920         BNEQ    40$                             ; Exit loop if no match
   F8 50   F5        004D   921         SOBGTR  R0,30$                          ; Any more bytes in S1?
                     0050   922
      EE   11        0050   923         BRB     IDENTICAL                       ; Exit indicating IDENTICAL strings
                     0052   924
54   71    91        0052   925 40$:    CMPB    -(R1),R4                        ; Reset R1 and set condition codes
      17   11        0055   926         BRB     NO_MATCH                        ; Exit indicating strings DO NOT MATCH
                     0057   927
                     0057   928 ; The following code executes if S1 has zero length on input. If S2 also
                     0057   929 ; has zero length, the routine smply returns, indicating equal strings.
                     0057   930
52   52    3C        0057   931 50$:    MOVZWL  R2,R2                           ; Clear unused bits. Is S2 len also zero?
      E4   13        005A   932         BEQL    IDENTICAL                       ; Exit indicating IDENTICAL strings
                     005C   933
                     005C   934 ; The following loop is entered when all of S1 has been processed but
                     005C   935 ; there are characters remaining in S2. In other words,
                     005C   936 ;
                     005C   937 ;       R0 EQL 0
                     005C   938 ;       R2 GTRU 0
                     005C   939 ;
                     005C   940 ; The remaining characters in S2 are compared to the fill character.
                     005C   941
                     005C   942         MARK_POINT      CMPC5_3
83   54    91        005C   943 60$:    CMPB    R4,(R3)+                        ; Characters match?
      05   12        005F   944         BNEQ    70$                             ; Exit loop if no match
   F8 52   F5        0061   945         SOBGTR  R2,60$                          ; Any more bytes in S2?
                     0064   946
      DA   11        0064   947         BRB     IDENTICAL                       ; Exit indicating IDENTICAL strings
                     0066   948
73   54    91        0066   949 70$:    CMPB    R4,-(R3)                        ; Reset R3 and set condition codes
      03   11        0069   950         BRB     NO_MATCH                        ; Exit indicating strings DO NOT MATCH
                     006B   951
                     006B   952 ; The following exit path is taken if both strings have characters
                     006B   953 ; remaining and a character pair that did not match was detected.
                     006B   954
73   71    91        006B   955 80$:    CMPB    -(R1),-(R3)                     ; Reset R1 and R3 and set condition codes
                     006E   956 NO_MATCH:                                       ; Restore R4 and R10
0410 8F    BA        006E   957         POPR    #^M<R4,R10>                     ;  without changing condition codes
```

```
05  0072   958          RSB                                 ; Exit indicating strings DO NOT MATCH
    0073   959
    0073   960          .DISABLE        LOCAL_BLOCK
```

```
                        0073  1152          .SUBTITLE       VAX$LOCC - Locate Character
                        0073  1153 ;+
                        0073  1154 ; Functional Description:
                        0073  1155 ;
                        0073  1156 ;       The character operand is compared with the bytes of the string specified
                        0073  1157 ;       by the length and address operands.  Comparison continues until equality
                        0073  1158 ;       is detected or all bytes of the string have been compared.  If  equality
                        0073  1159 ;       is  detected;  the condition code Z-bit is cleared;  otherwise the Z-bit
                        0073  1160 ;       is set.
                        0073  1161 ;
                        0073  1162 ; Input Parameters:
                        0073  1163 ;
                        0073  1164 ;       R0<15:0>  = len         Length of character string
                        0073  1165 ;       R0<23:16> = char        Character to be located
                        0073  1166 ;       R1        = addr        Address of character string
                        0073  1167 ;
                        0073  1168 ; Intermediate State:
                        0073  1169 ;
                        0073  1170 ;     31                23                15                07              00
                        0073  1171 ;     +-----------------+-----------------+-----------------+---------------+
                        0073  1172 ;     |    delta-PC     |      char       |                 len             | : R0
                        0073  1173 ;     +-----------------+-----------------+-----------------+---------------+
                        0073  1174 ;     |                                   addr                              | : R1
                        0073  1175 ;     +-----------------+-----------------+-----------------+---------------+
                        0073  1176 ;
                        0073  1177 ; Output Parameters:
                        0073  1178 ;
                        0073  1179 ;       Character Found
                        0073  1180 ;
                        0073  1181 ;               R0 = Number of bytes remaining in the string (including located one)
                        0073  1182 ;               R1 = Address of the located byte
                        0073  1183 ;
                        0073  1184 ;       Character NOT Found
                        0073  1185 ;
                        0073  1186 ;               R0 = 0
                        0073  1187 ;               R1 = Address of one byte beyond end of string
                        0073  1188 ;
                        0073  1189 ; Condition Codes:
                        0073  1190 ;
                        0073  1191 ;       N <- 0
                        0073  1192 ;       Z <- R0 EQL 0
                        0073  1193 ;       V <- 0
                        0073  1194 ;       C <- 0
                        0073  1195 ;
                        0073  1196 ;       The Z bit is clear if the character is located.
                        0073  1197 ;       The Z bit is set if the character is NOT located.
                        0073  1198 ;
                        0073  1199 ; Side Effects:
                        0073  1200 ;
                        0073  1201 ;       This routine uses two longwords of stack
                        0073  1202 ;-
                        0073  1203
                        0073  1204 VAX$LOCC::
5A   DD                 0073  1205          PUSHL   R10                     ; Save R10 so it can hold handler
                        0075  1206          ESTABLISH_HANDLER       -
                        0075  1207                  STRING_ACCVIO           ; Store address of condition handler
52   DD                 0075  1208          PUSHL   R2                      ; Save register
```

```
                     52   50    F0 8F    78   0077  1209          ASHL    #-16,R0,R2              ; Get character to be located
                          50       50    3C   007C  1210          MOVZWL  R0,R0                   ; Clear unused bits & check for 0 length
                                   08    13   007F  1211          BEQL    20$                     ; Simply return if length is 0
                                               0081  1212
                                               0081  1213          MARK_POINT      LOCC_1
                                81     52    91   0081  1214 10$:     CMPB    R2,(R1)+                ; Character match?
                                       0A    13   0084  1215          BEQL    30$                     ; Exit loop if yes
                                     F8 50   F5   0086  1216          SOBGTR  R0,10$
                                               0089  1217
                                               0089  1218 ; If we drop through the end of the loop into the following code, then
                                               0089  1219 ; the input string was exhausted with the character NOT found.
                                               0089  1220
                                0404   8F     BA   0089  1221 20$:     POPR    #^M<R2,R10>             ; Restore saved R2 and R10
                                       50     D5   008D  1222          TSTL    R0                      ; Insure that C-bit is clear
                                              05   008F  1223          RSB                             ; Return with Z-bit set
                                               0090  1224
                                               0090  1225 ; Exit path when character located
                                               0090  1226
                                       51     D7   0090  1227 30$:     DECL    R1                      ; Point R1 to located character
                                       F5     11   0092  1228          BRB     20$                     ; Join common code
```

```
                                                I 13
0094  2168          END_MARK_POINT
0094  2169
0094  2170          .END
```

```
BOOT_SWITCH                     = 00000001
IDENTICAL                         00000040 R      02
NO_MATCH                          0000006E R      02
OP$_ACBD                        = 0000006F
OP$_ACBF                        = 0000004F
OP$_ACBG                        = 00004FFD
OP$_ACBH                        = 00006FFD
OP$_ADDD2                       = 00000060
OP$_ADDD3                       = 00000061
OP$_ADDF2                       = 00000040
OP$_ADDF3                       = 00000041
OP$_ADDG2                       = 000040FD
OP$_ADDG3                       = 000041FD
OP$_ADDH2                       = 000060FD
OP$_ADDH3                       = 000061FD
OP$_ADDP4                       = 00000020
OP$_ADDP6                       = 00000021
OP$_ASHP                        = 000000F8
OP$_CLRD                        = 0000007C
OP$_CLRF                        = 000000D4
OP$_CLRG                        = 0000007C
OP$_CLRH                        = 00007CFD
OP$_CMPD                        = 00000071
OP$_CMPF                        = 00000051
OP$_CMPG                        = 000051FD
OP$_CMPH                        = 000071FD
OP$_CMPP3                       = 00000035
OP$_CMPP4                       = 00000037
OP$_CRC                         = 0000000B
OP$_CVTBD                       = 0000006C
OP$_CVTBF                       = 0000004C
OP$_CVTBG                       = 00004CFD
OP$_CVTBH                       = 00006CFD
OP$_CVTDB                       = 00000068
OP$_CVTDF                       = 00000076
OP$_CVTDH                       = 000032FD
OP$_CVTDL                       = 0000006A
OP$_CVTDW                       = 00000069
OP$_CVTFB                       = 00000048
OP$_CVTFD                       = 00000056
OP$_CVTFG                       = 000099FD
OP$_CVTFH                       = 000098FD
OP$_CVTFL                       = 0000004A
OP$_CVTFW                       = 00000049
OP$_CVTGB                       = 000048FD
OP$_CVTGF                       = 000033FD
OP$_CVTGH                       = 000056FD
OP$_CVTGL                       = 00004AFD
OP$_CVTGW                       = 000049FD
OP$_CVTHB                       = 000068FD
OP$_CVTHD                       = 0000F7FD
OP$_CVTHF                       = 0000F6FD
OP$_CVTHG                       = 000076FD
OP$_CVTHL                       = 00006AFD
OP$_CVTHW                       = 000069FD
OP$_CVTLD                       = 0000006E
OP$_CVTLF                       = 0000004E
OP$_CVTLG                       = 00004EFD
OP$_CVTLH                       = 00006EFD
OP$_CVTLP                       = 000000F9
OP$_CVTPL                       = 00000036
OP$_CVTPS                       = 00000008
OP$_CVTPT                       = 00000024
OP$_CVTRDL                      = 0000006B
OP$_CVTRFL                      = 0000004B
OP$_CVTRGL                      = 00004BFD
OP$_CVTRHL                      = 00006BFD
OP$_CVTSP                       = 00000009
OP$_CVTTP                       = 00000026
OP$_CVTWD                       = 0000006D
OP$_CVTWF                       = 0000004D
OP$_CVTWG                       = 00004DFD
OP$_CVTWH                       = 00006DFD
OP$_DIVD2                       = 00000066
OP$_DIVD3                       = 00000067
OP$_DIVF2                       = 00000046
OP$_DIVF3                       = 00000047
OP$_DIVG2                       = 000046FD
OP$_DIVG3                       = 000047FD
OP$_DIVH2                       = 000066FD
OP$_DIVH3                       = 000067FD
OP$_DIVP                        = 00000027
OP$_EDITPC                      = 00000038
OP$_EMODD                       = 00000074
OP$_EMODF                       = 00000054
OP$_EMODG                       = 000054FD
OP$_EMODH                       = 000074FD
OP$_MATCHC                      = 00000039
OP$_MNEGD                       = 00000072
OP$_MNEGF                       = 00000052
OP$_MNEGG                       = 000052FD
OP$_MNEGH                       = 000072FD
OP$_MOVD                        = 00000070
OP$_MOVF                        = 00000050
OP$_MOVG                        = 000050FD
OP$_MOVH                        = 000070FD
OP$_MOVP                        = 00000034
OP$_MOVTC                       = 0000002E
OP$_MOVTUC                      = 0000002F
OP$_MULD2                       = 00000064
OP$_MULD3                       = 00000065
OP$_MULF2                       = 00000044
OP$_MULF3                       = 00000045
OP$_MULG2                       = 000044FD
OP$_MULG3                       = 000045FD
OP$_MULH2                       = 000064FD
OP$_MULH3                       = 000065FD
OP$_MULP                        = 00000025
OP$_POLYD                       = 00000075
OP$_POLYF                       = 00000055
OP$_POLYG                       = 000055FD
OP$_POLYH                       = 000075FD
OP$_SCANC                       = 0000002A
OP$_SKPC                        = 0000003B
```

```
OP$_SPANC                       = 0000002B
OP$_SUBD2                       = 00000062
OP$_SUBD3                       = 00000063
OP$_SUBF2                       = 00000042
OP$_SUBF3                       = 00000043
OP$_SUBG2                       = 000042FD
OP$_SUBG3                       = 000043FD
OP$_SUBH2                       = 000062FD
OP$_SUBH3                       = 000063FD
OP$_SUBP4                       = 00000022
OP$_SUBP6                       = 00000023
OP$_TSTD                        = 00000073
OP$_TSTF                        = 00000053
OP$_TSTG                        = 000053FD
OP$_TSTH                        = 000073FD
VAX$CMPC3                         00000000 RG    02
VAX$CMPC5                         00000021 RG    02
VAX$LOCC                          00000073 RG    02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_VAX$CODE	00000094 (148.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	15	00:00:00.06	00:00:01.22
Command processing	74	00:00:00.73	00:00:05.99
Pass 1	390	00:00:11.56	00:00:41.58
Symbol table sort	0	00:00:00.58	00:00:01.86
Pass 2	102	00:00:05.40	00:00:15.24
Symbol table output	16	00:00:00.11	00:00:00.40
Psect synopsis output	2	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	599	00:00:18.45	00:01:06.31

```
The working set limit was 1500 pages.
70465 bytes (138 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 447 non-local and 14 local symbols.
4923 source lines were read in Pass 1, producing 13 object records in Pass 2.
145 pages of virtual memory were used to define 143 macros.
```

```
                                +----------------------------+
                                ! Macro library statistics !
                                +----------------------------+

Macro library name                                Macros defined
------------------                                --------------
_$255$DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1                        8
_$255$DUA28:[SYSLIB]STARLET.MLB;2                              5
TOTALS (all libraries)                                        13

584 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:BOOSTRING/OBJ=OBJ$:BOOSTRING MSRC$:BOOTSWT/UPDATE=(ENH$:BOOTSWT)+MSRC$:MISSING/UPDATE=(ENH$:MISSING)+MSRC$:VAXSTRING/
```

XLATE
LIS

VAXEMUL
MAP

WIOHNDLR
LIS

WORKIO
LIS

BOOSTRING
LIS

EMULAT

FPEMUL
MAP

BOOEMULAT
LIS

WORDWRAP
LIS

VAXMACROS
MAR

WRIEDTMSG
LIS

FPEMULATE
LIS

VAXREGDEF
SDL